SAFIRE Systems Engineering Tutorial

Version 19.02+

Powered by SAFIRE. Pure Signaling.

SOLiNET Solutions for Innovative Networks!

SAFIRE Systems Engineering Tutorial

Version 19.02+

Copyright © 1990-2007 SOLINET GmbH.

No part of this document may be reproduced in any form without written permission from SOLINET GmbH.

Specifications subject to change without notice. All product and company names are trademarks of their respective owners. All rights reserved.

SOLINET GmbH
Mittlerer Pfad 26
70499 Stuttgart, Germany

Tel +49 (0) 711 1398 0
Fax +49 (0) 711 866 1240
Email Info@SOLINET.com
Web www.SOLINET.com

For technical support please contact: Support@SOLINET.com

Table of Contents

Introduction 5
Overview 5
Prerequisites 5
SBUS Application Design 6
Requirements 6
System Overview 7
Create a Project 10
Start the SAFIRE Organizer 10
Create and Edit a System Project 10
Create an Interface 11
Create Stub FSMs 12
Create Process Types 12
Create Gates for Master_T and Slave_T 12
Create Gates for SBUS_T 13
Create and Connect Instances 14
Create FSM Instances 14
Create Connections 14
SBUS Behavior 15
Create SBUS States 15
Define SBUS Start Transition 15
Create First SBUS Idle State Input 16
Create Second SBUS Idle State Input 16
Implement SBUS Active State 18
Slave FSM Type 19
Create States 19
Set Start Transition 19
Implement Idle State 19
Implement Active State 20
Master FSM Type 21
Create States 21

SAFIRE Sytems Engineering Tutorial
Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

Set Start Transition 21
Implement Act State 21
Implement Data State 22
Implement Deact State 23
Error Detection 24
Syntax Errors 24
Semantic Errors 24
Running the SBUS Application 25
Build Projects 25
Run the System 25
View Messages in the System Tracer 26
Shut Down the System 27

SAFIRE Sytems Engineering Tutorial
Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

INTRODUCTION

OVERVIEW

This Tutorial will guide you step by step in the creation and execution of a Finite State Machine (FSM) called the SBUS. Although the SBUS is simple, this tutorial will give you a chance to use all of the core tools of the SAFIRE environment.

First you will create a new system project in the SAFIRE Organizer, and create the SBUS FSM in the SAFIRE Editor. To exercise the SBUS, you will create two other FSMs, Master and Slave, within the same system. The Master and Slave processes will send signals to each other through the SBUS. After compiling in the SAFIRE Organizer, you can run the new system in the SAFIRE environment, observing it with the SAFIRE Tracer and SAFIRE Animator.

PREREQUISITES

The tutorial requires familiarity with Windows and programming. It does not require previous knowledge of SAFIRE tools, the SDL language, or finite state machine terminology.

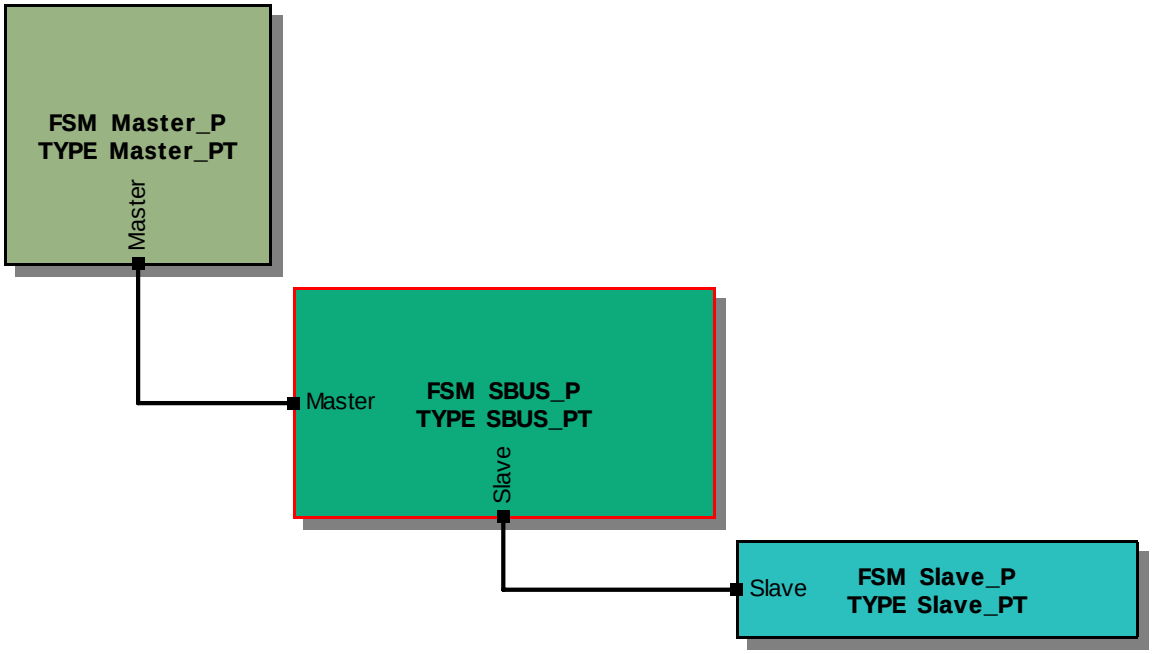
The tutorial assumes that the SAFIRE software and license files have been installed on your computer, and the appropriate software-sharing device has been attached. For help in resolving installation issues, please consult the Installation Manual.

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

SBUS APPLICATION DESIGN

REQUIREMENTS

Suppose there are two devices, Master and Slave, simulated as finite state machines (FSMs) in SAFIRE. You wish to connect the two FSMs. The real Master and Slave devices would be connected by a cable. To simulate the cable, a third FSM, the SBUS, should be created.



Structure of the SBUS system

The Master and Slave devices send "Request" signals.

Act_Rq	Activate Request
Deact_Rq	Deactivate Request
Data_Rq	Data Request

The Master and Slave devices receive "Indication" signals.

SAFIRE Sytems Engineering Tutorial
Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

Act_Ind	Activate Indication
Deact_Ind	Deactivate Indication
Data_Ind	Data Indication

Both devices send data by sending a Data Request signal, and receive data by receiving a Data Indication signal.

Before the devices begin transferring data, one of them must first send an Activate Request signal then both of them must receive an Activate Indication signal as confirmation that data transfer can begin.

At any time, either device can end data transfer by sending a Deactivate Request signal, which should be confirmed by a Deactivate Indication signal to both.

A direct connection between the Master and Slave will not work. Both processes expect to receive Indication signals, but cannot send them, and expect to send Request signals, but cannot receive them. As a connector, a third device is needed to translate Requests into Indications and to send Activate and Deactivate Indications to both devices. This third device, a “software bus”, will be called the SBUS.

System Overview

System Project

The first project of this tutorial is a system project, a top-level project that can be executed in the SAFIRE environment. The system will contain three processes: an SBUS, a Master, and a Slave. The Master and Slave processes will be connected to the SBUS.

SBUS Process

The SBUS process will have two states, Idle and Active. The initial state will be Idle.

At any time, upon receiving an Activate Request signal from either side, the SBUS will send an Activate Indication signal to both sides, and sets its own state to Active.

SAFIRE Sytems Engineering Tutorial
Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

At any time, upon receiving a Deactivate Request signal from either side, the SBUS will send a Deactivate Indication signal to both sides, and sets its own state to Idle.

In the Active state, upon receiving a Data Request signal from one side, the SBUS will send a Data Indication signal to the other side. In the first version of the SBUS simulation, for simplicity's sake there will be no actual data in the data signals. In more advanced SBUS versions, we will enhance the data signals with a parameter, an octet string carrying arbitrary data.

MASTER PROCESS

The Master process cycles from one state to another in a simple fashion, sending a signal, then entering a new state in which it awaits the next signal.

At the start, the Master process will send an Activate Request signal then enter the Act state.

In the Act state, the Master process will wait for an Activate Indication signal, then send a Data Request signal and enter the Data state.

In the Data state, the Master process will wait for a Data Indication signal, then send a Deactivate Request signal and enter the Deact state.

In the Deact state, the Master process will wait for a Deactivate Indication signal, then start the cycle again by sending an Activate Request signal and entering the Act state.

SLAVE PROCESS

At the start, the Slave process will enter the Idle state.

In the Idle state, the Slave process will wait for an Activate Indication signal, then enter the Active state.

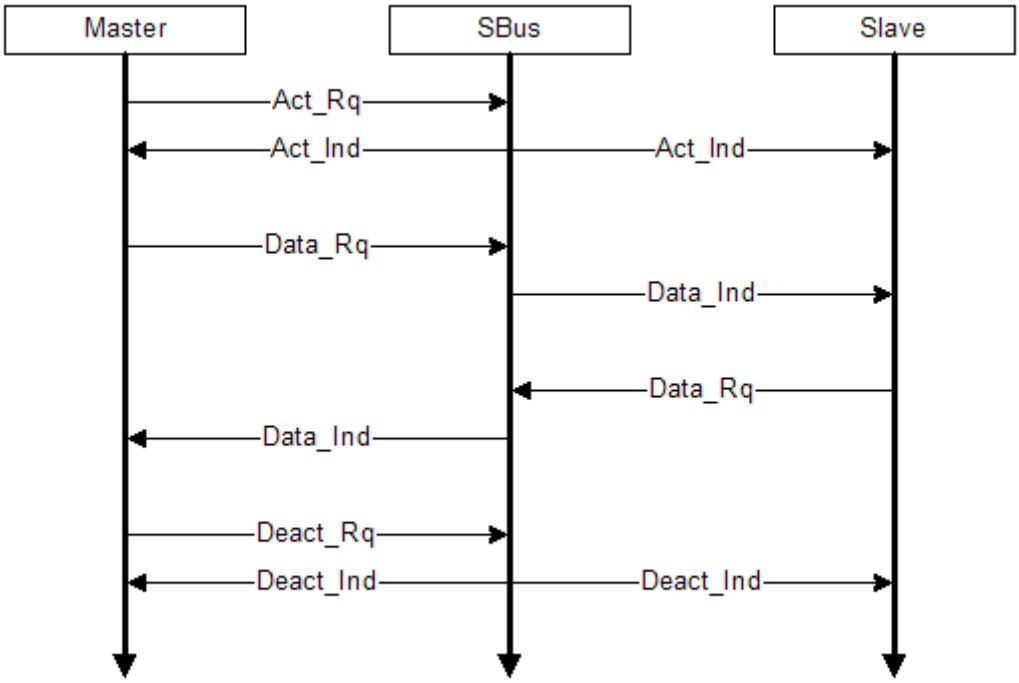
In the Active state, the Slave process will wait for two signals:

If a Deactivate Indication signal arrives, the Slave process will enter the Idle state.

If a Data Indication signal arrives, the Slave process will send a Data Request signal and enter the Active state.

The flow of signals among the three processes is summarized in the following message sequence chart.

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.



Message sequence chart for the SBUS system

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

CREATE A PROJECT

Before creating state machines, a SAFIRE archive must be created to contain the internal code. A SAFIRE system project will be created inside the archive.

START THE SAFIRE ORGANIZER

All SAFIRE tools are launched from the Organizer, including the Editor and the compiler, so the Organizer must be launched first.

1. Select Programs > SAFIRE Professional nnnnnnnn > SAFIRE Organizer (where "nnnnnnnn" is the appropriate SAFIRE version number) from the Windows Start menu.
2. In the Select an archive dialog, click the Create button. The Create New Archive dialog will appear.
3. Enter "Tutorial" as the archive name.
4. Enter "C:\" (or any other path you wish) as the archive path.
5. Set the build mode to "SAFIRE Only", because this tutorial does not require any C code.
6. Set the user mode to "Single User Mode".
7. Click OK. The Create New Version dialog will appear.
8. Leave all fields blank and click OK. The Organizer will create the archive and open it.

CREATE AND EDIT A SYSTEM PROJECT

A system project can be executed. For the sake of simplicity in this tutorial, a unique code will be created within a single system project.

1. Right-click on the ROOT folder and select Create > System from the context menu.
2. In the Create New System dialog, enter "SBUS1_APPL" in the Name field.
3. Select the new SBUS1_APPL system and click the Edit button in the Organizer toolbar. The project will be opened in a SAFIRE Editor window.

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

CREATE AN INTERFACE

The Requirements section above describes six signals that can be sent or received by an SBUS.

Act_Rq	Request from upper layer to SBUS:	Activate
Data_Rq	"	Send data packet
Deact_Rq	"	Deactivate
Act_Ind	Indication from SBUS to upper layer:	Activated
Data_Ind	"	Data packet received
Deact_Ind	"	Deactivated

In SAFIRE, a group of signals is called an interface. In an FSM, each gate is assigned a specific interface, and a gate can send and receive only the signals listed in its interface. Two gates may be connected only if they have the same interface.

In this step of the tutorial an interface containing these six signals will be created first. Later this interface will be assigned to the gates of our FSM types.

1. In the Editor, right-click on the INTERFACES node of the SBUS1_APPL system and select Insert > INTERFACE from the context menu. Name the new interface “SBUS_INTF”.
2. Right-click on the SBUS_INTF interface node and select Insert > SIGNAL from the context menu. In the Name field, enter the signal name "Act_Rq".
3. In the same manner, create the other five signals within SBUS_INTF.
4. Save changes.

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

CREATE STUB FSMS

Following a top-down development style, three top-level FSM types should be created for use in the system.

In this section, the "stub" FSM types are created. They are complete on the outside but empty on the inside. Later in the document the internal states and behavior will be specified.

CREATE PROCESS TYPES

A process type for each of our 3 processes should be created. Later, after defining other parts of the system, return to the process types to define their state behavior.

1. Right click on FSM TYPES and select Insert > FSM Definition. Name the new type "Master_T".
2. Create two more process types named Slave_T and SBUS_T.
3. Save changes.

CREATE GATES FOR MASTER_T AND SLAVE_T

The only way to send a signal from one FSM to another is via a channel. A channel connects two gates that have the same interface. An interface has already been defined, so now the gates must be defined.

Master_T and Slave_T need one gate each, leading "down" to the FSM connecting them. Note that same gate name can be used in two different FSM types.

1. Edit Master_T by double-clicking the FSM Master_T node in the FSM SBUS1_APPL structure tree.
2. In the Master_T window, right-click on the GATES node and select Insert > GATE.
3. Name the new gate “Lower”, because it will send signals to a lower layer (in this case, the SBUS). From the dropdown list, select the SBUS_INTF interface.
4. Save changes.
5. Return to the system by clicking the "Up" button in the toolbar.
6. Edit Slave_T by double-clicking on it.
7. Add a gate to the Slave_T process type. Name the gate "Lower" and select SBUS_INTF as its interface.
8. Save changes.
9. Return to the system by clicking the "Up" button in the toolbar.

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

CREATE GATES FOR SBUS_T

SBUS_T needs two gates, one for each FSM connected to it.

1. Edit SBUS_T by double-clicking on it.
2. Add two gates to the SBUS_T process type. Name the gates "Upper_1" and "Upper_2" and select SBUS_INTF as the interface for both gates.
3. Save changes.

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

CREATE AND CONNECT INSTANCES

The FSM types are ready to be instantiated and connected.

CREATE FSM INSTANCES

For each FSM type a single instance is required.

1. Within the BEHAVIOR BLOCK node, right click on FSM INSTANCES and select Insert > INSTANCE.
2. Enter “Master” in the Name field, select Master_T in the FSM Type list, and click OK.
3. Create instances of the other two FSM types (Slave of type Slave_T and SBUS of type SBUS_T).
4. Save changes.

CREATE CONNECTIONS

Now connect the Master and Slave instances to the SBUS.

1. Right-click on the CONNECTIONS in the Editor tree view and select Insert > Connection from the context menu. The Connection dialog appears, with fields for selecting the instances and gates to be connected.
2. On the left side, select Master as the first instance. Lower will be selected automatically as the gate.
3. On the right side, select SBUS as the second endpoint. Upper_1 will be selected automatically as the gate.
4. Add another connection that connects the Lower gate of the Slave FSM to the Upper_2 gate of the SBUS FSM.
5. Save changes.

The high-level structure of the tutorial application is now complete.

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

SBUS BEHAVIOR

In this section states within the FSM types will be created. Within each state a flow diagram will be created to describe the FSM's behavior upon receiving signals.

CREATE SBUS STATES

1. In the FSM Types view on the left side of the Editor, open the SBUS_T node by double-clicking it.
2. Right-click on the STATES node and click Insert > STATE. Name the new state “Idle”.
3. Create a second new state named “Active”.

DEFINE SBUS START TRANSITION

The START transition of the SBUS should put the SBUS in the Idle state.

4. Right-click on the START symbol and click Insert > NEXTSTATE.
5. In the NEXT STATE Editor, select the Idle state and click OK. The start transition is now complete.
6. Save changes.

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

CREATE FIRST SBUS IDLE STATE INPUT

In the Idle state, the SBUS should respond to an incoming Act_Rq signal by sending Act_Ind to Master and Slave and then changing its state to Active.

1. To go to the Idle state, select the Idle state in the tree view. Alternately, right-click on the START state's NEXTSTATE symbol and select Go To Definition from the context menu).
2. Right-click on the Idle state node and select Insert > INPUT from the context menu.
3. In the Input dialog, set the signal name to Act_Rq, set the gate name to ANY and click OK.
4. Right-click on the INPUT Act_Rq node and select Insert > OUTPUT from the context menu.
5. In the Output dialog, click Signal, select Act_Ind in the Signal dropdown list, select Upper_1 in the Gate dropdown list, and click OK.
6. Add another output symbol after the first one, sending the Act_Ind signal to the Upper_2 gate.
7. Right-click on the second output symbol and select Insert After > NEXTSTATE from the context menu.
8. Select the Active state as the destination and click OK.
9. Save changes.

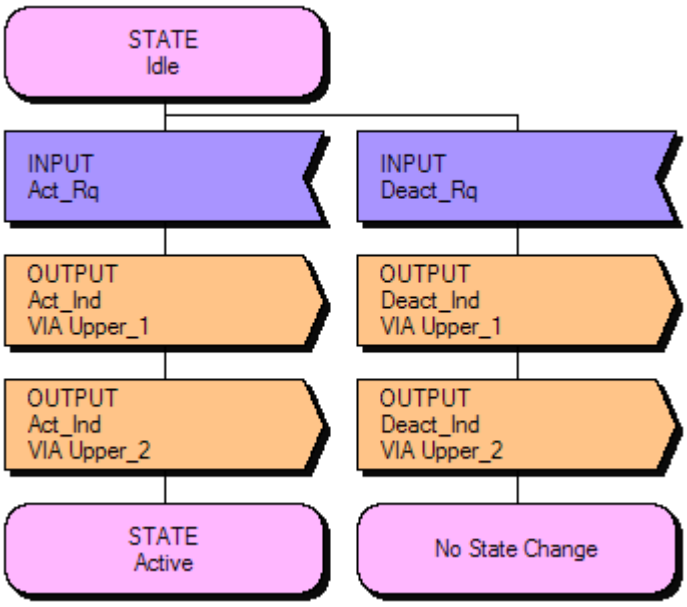
CREATE SECOND SBUS IDLE STATE INPUT

In the Idle state, an incoming Deact_Rq signal requires sending Deact_Ind to Master and Slave without changing the state of the SBUS. The new input handler should be at the same level as the first input handler that you just created.

1. Right-click on the existing Act_Rq input symbol and select Insert Right > INPUT from the context menu.
2. In the Input dialog, set the signal name to Deact_Rq and the gate name to ANY.
3. Insert an output symbol under the new input, sending Deact_Ind to the Upper_1 gate.
4. Insert another output symbol after the first, sending Deact_Ind to Upper_2.
5. Insert a NEXTSTATE symbol with No State Change as its destination.
6. Save changes.

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

The Idle state should now look like the following screenshot:



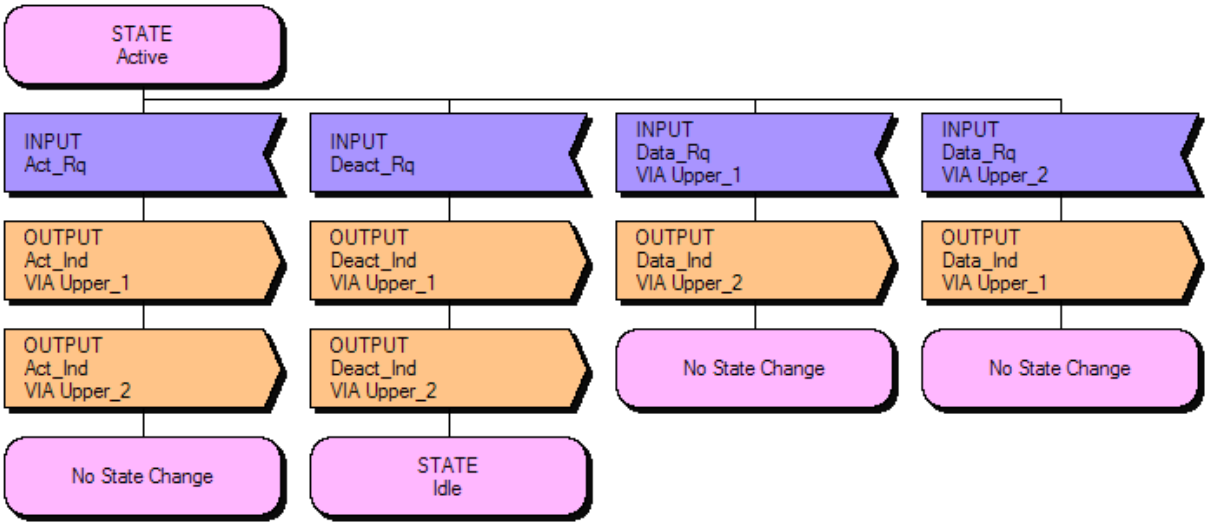
SBUS Idle state

SAFIRE Sytems Engineering Tutorial
Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

IMPLEMENT SBUS ACTIVE STATE

The Active state will handle Activate and Deactivate signals and will also pass Data signals from one gate to the other. Reuse the input handlers from the Idle state, making minor changes.

1. Copy the Idle state's input nodes and paste them into the Active state.
2. In the Active state, reset the NEXTSTATE for both inputs. Below the Act_Rq input, there should be no state change. Below the Deact_Rq input, there should be a transition to the Idle state.
3. Insert a third input symbol to the right of the existing inputs. In the Input dialog, set the signal name to Data_Rq and the gate name to Upper_1.
4. Right-click on the new input symbol and add an output symbol. In the Output dialog, set the signal name to Data_Ind. Set the gate name to Upper_2.
5. Below the new output symbol add a NEXTSTATE symbol with the value No State Change.
6. Right-click on the new input symbol and select Duplicate from the context menu. A copy of the input symbol and its action list will be added to the flow diagram.
7. Edit the duplicate input, setting the gate to Upper_2.
8. Edit the duplicate output, setting the gate to Upper_1.



SBUS_T Active state

The SBUS_T FSM type is now complete. Save the project and return to the system view.

SAFIRE Sytems Engineering Tutorial
Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

SLAVE FSM TYPE

The Slave FSM type is simple. It exists merely to exercise the SBUS.

CREATE STATES

The Slave FSM type will need two states, Idle and Active.

1. Double click the Slave_T FSM type to edit it.
2. Add the two new states.

SET START TRANSITION

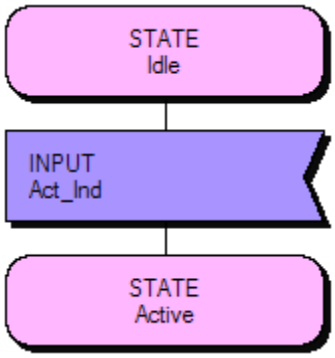
At system startup, the Slave FSM should transition immediately to the Idle state.

1. Insert a NEXTSTATE symbol into the Start transition and set it to “Idle”.

IMPLEMENT IDLE STATE

The only signal handled in the Idle state is the Act_Ind signal, which should cause a transition to the Active state.

1. Insert an INPUT into the Idle state, with the signal name set to Act_Ind. There is only one gate in this FSM type, so the default ANY GATE option is OK.
2. Within the new input set the next state to Active.



Slave_T Idle state

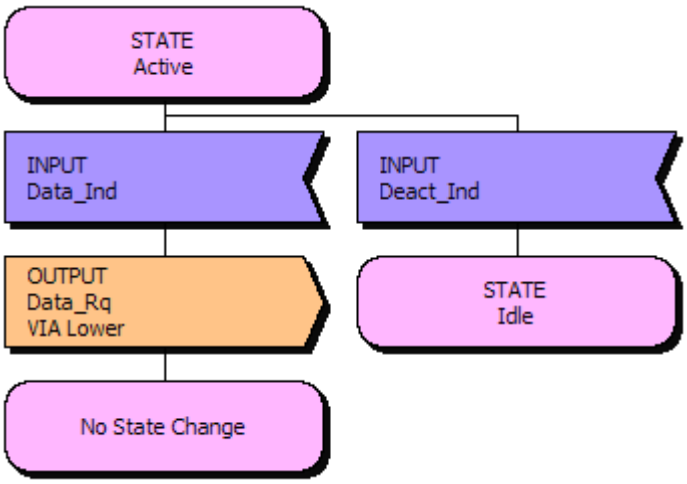
Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

IMPLEMENT ACTIVE STATE

In the Active state, the Slave should reply to an incoming data indication by sending a data request. The Slave should remain in the Active state, because this behavior should recur until the Slave is deactivated.

In response to a Deact_Ind signal, the Slave should enter the Idle state.

1. In the Active state, insert an input symbol for the Data_Ind signal.
2. After the new input symbol, insert a new output symbol, sending the Data_Rq signal via the Lower gate.
3. After the output, put a NEXTSTATE symbol at the end of the action list, with the value "No State Change..
4. Insert another input (to the right of the first input) for the Deact _Ind signal.
5. In the action list of the new Deact_Ind input, create a transition to the Idle state.



Slave_T Active state.

The Slave_T FSM type is now complete. Save the project and return to the system view.

SAFIRE Sytems Engineering Tutorial
Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

MASTER FSM TYPE

The Master FSM will start the tutorial action by sending the first signal. Each response from the SBUS will trigger another signal output from the Master, causing the application to run in an infinite loop.

CREATE STATES

The Master FSM type will need three states: Act, Data and Deact. The state names reflect the name of the signal expected in each state.

1. Double click the Master_T FSM type to edit it.
2. Add the three new states.

SET START TRANSITION

At system startup, the Master FSM should send an Act_Rq signal then transition to the Act state.

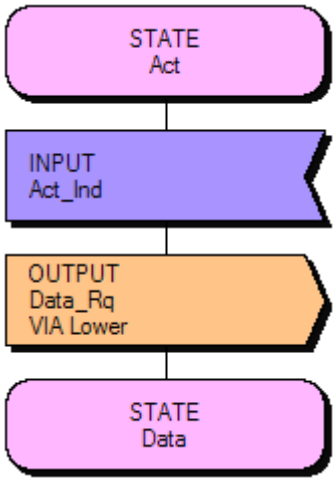
1. Insert an output under the START symbol, sending the Act_Rq signal via the Lower gate.
2. Insert a NEXTSTATE symbol after the output symbol and set it to “Act”.

IMPLEMENT ACT STATE

The only signal handled in the Act state is the Act_Ind signal, which should cause a transition to the Data state.

1. Insert an INPUT into the Act state, with the signal name set to Act_Ind. There is only one gate in this FSM type, so the default ANY GATE option is OK.
2. Insert an OUTPUT under the input symbol. Send the Data_Rq signal via the Lower gate.
3. At the end of the new input handler set the next state to Data. See screenshot below.

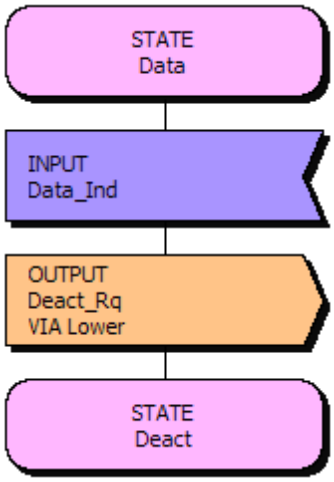
Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.



Master_T Act state

IMPLEMENT DATA STATE

The Data state is similar to the Act state. The input handler accepts the Data_Ind signal, sends the Deact_Rq signal, and transitions to the Deact state.



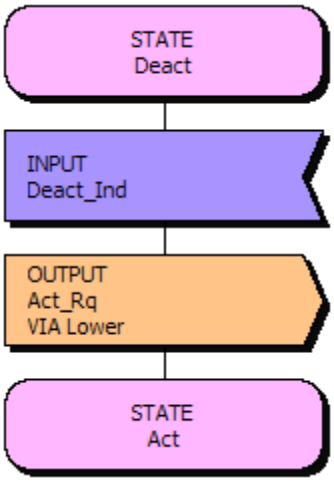
Master_T Data state

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

IMPLEMENT DEACT STATE

The Deact state is also similar to the Act state. The input handler accepts the Deact_Ind signal, sends the Act_Rq signal, and transitions to the Act state.

This forms an endless loop. At runtime, this endless loop can be halted by pausing the SAFIRE runtime (using the Pause command in the File menu).



Master_T Deact state

The Master_T FSM type is now complete. Please Save.

SAFIRE Sytems Engineering Tutorial
Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

ERROR DETECTION

SYNTAX ERRORS

If the source file was modified outside the Editor, or an old file has been opened that is no longer accepted by the latest SAFIRE version, the Editor might not be able to read the file.

Also, in certain circumstances it is possible to create an incomplete specification in the Editor. If the Editor cannot parse the source file, it will refuse to open the file. Instead, it will offer to attempt to build the offending project in the Organizer. The Organizer output window will display the filename and line number where the syntax error is found.

After correcting the syntax error in a text editor, reopen the project to check it again.

SEMANTIC ERRORS

If there is a semantic error in the program, the SAFIRE compiler will not be able to compile it. In this case, the output window of the Organizer will display a list of compilation errors. Double-clicking on an error message will open the file in the Editor at the location of the error.

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

RUNNING THE SBUS APPLICATION

BUILD PROJECTS

Before executing the system, the projects must be compiled for the SAFIRE Runtime.

1. Switch to the Organizer application.
2. Select the SBUS1_APPL system project.
3. Click build button in the toolbar, or right-click on the project and select build.
4. In the Build Options dialog, leave all the options unchecked and click OK.
5. The Organizer will compile the project and report success or failure in the output window at the bottom of the application window
6. The status icons next to the SBUS1_APPL project should turn green, and in the project list on the right side the status should change from "Out-of-date" to "OK".

Status icons before (on left) and after (on right) a successful build

If the build failed, read the error message in the output window and consult the Error Detection section of this document.

RUN THE SYSTEM

From the Organizer the compiled system project can be launched in the SAFIRE Runtime.

1. Select the SBUS1_APPL system project in the Organizer.
2. Click run button in the toolbar, or right-click on the project and select Run.
3. In the Run dialog, check the Enable System Trace checkbox.
4. Leave the other checkboxes unchecked (for now) and click OK.

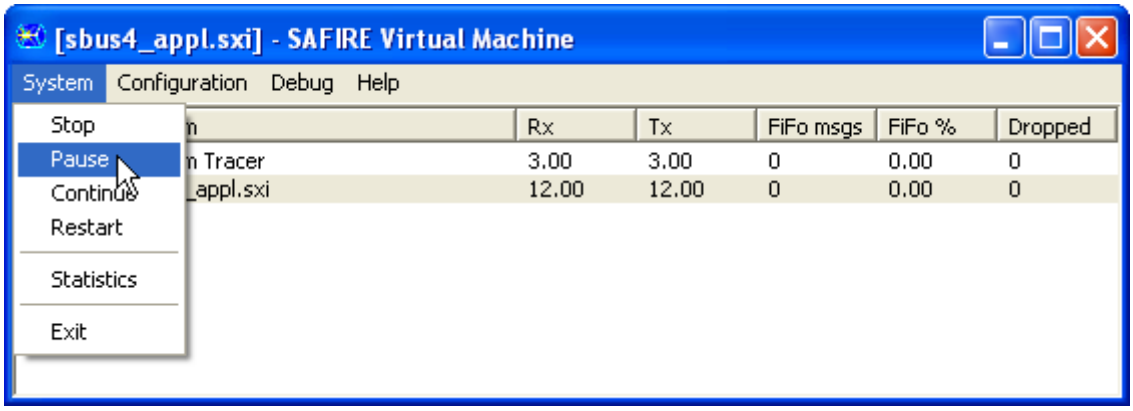
Two new application windows will appear: one for SAFIRE Virtual Machine and one for SAFIRE Tracer. The Virtual Machine window allows system configuration. The Tracer window displays messages representing the signals sent and received by each FSM in a given system.

Note that this system sends messages without pausing between them. Tens of thousands of messages per second may be sent, using all available CPU time. To restore normal computer operation, pause the system:

1. Restore the Virtual Machine window from the taskbar.

SAFIRE Sytems Engineering Tutorial
Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.

2. Select the SBUS1_APPL system in the Virtual Machine (run time) window. Note that the Tx and Rx columns (showing the number of signals transmitted and received per second) contain high values.
3. Select Pause from the System menu. The numbers in the Rx and Tx columns will drop to zero.



Pausing a system in the Virtual Machine window

The system execution can be continued later, if desired.

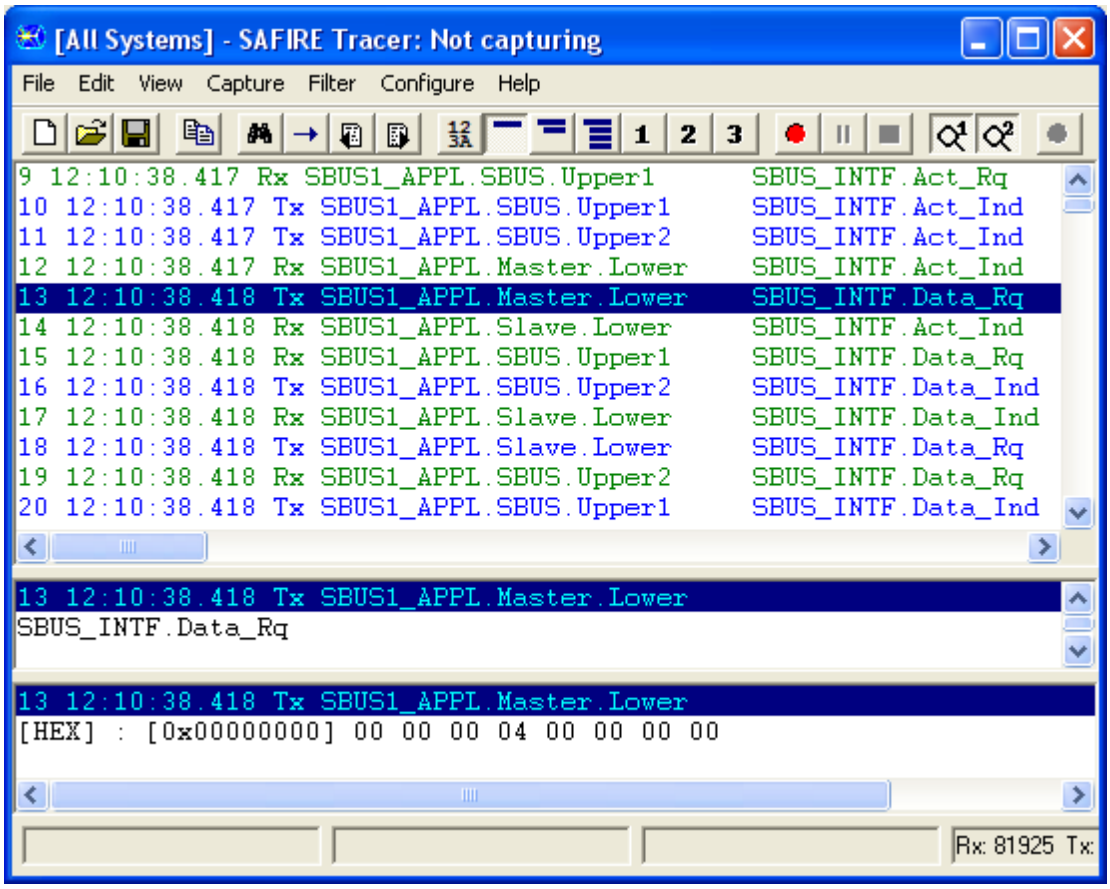
VIEW MESSAGES IN THE SYSTEM TRACER

The System Tracer shows all messages in the system. In SAFIRE, a "message" is the transmission or reception of a signal at an FSM gate. By default, a given signal will appear twice in the System Tracer, once when it is sent, and once more when it is received. The first message will have the keyword "Tx" in its header; the second will have "Rx".

In the Tracer, the capture of messages can be started and stopped independently of the system under observation. To stop capture, click the stop button in the toolbar. When the Tracer is stopped, one can scroll up and down in the window to look at any message. It is also possible to search and to apply a filter to captured messages. For more information, open the Tracer user manual by selecting Help > About in the Tracer window.

To restart capture, click the start capture button in the Tracer toolbar.

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.


[All Systems] - SAFIRE Tracer: Not capturing
File Edit View Capture Filter Configure Help
9 12:10:38.417 Rx SBUS1_APPL.SBUS.Upper1 SBUS_INTF.Act_Rq
10 12:10:38.417 Tx SBUS1_APPL.SBUS.Upper1 SBUS_INTF.Act_Ind
11 12:10:38.417 Tx SBUS1_APPL.SBUS.Upper2 SBUS_INTF.Act_Ind
12 12:10:38.417 Rx SBUS1_APPL.Master.Lower SBUS_INTF.Act_Ind
13 12:10:38.418 Tx SBUS1_APPL.Master.Lower SBUS_INTF.Data_Rq
14 12:10:38.418 Rx SBUS1_APPL.Slave.Lower SBUS_INTF.Act_Ind
15 12:10:38.418 Rx SBUS1_APPL.SBUS.Upper1 SBUS_INTF.Data_Rq
16 12:10:38.418 Tx SBUS1_APPL.SBUS.Upper2 SBUS_INTF.Data_Ind
17 12:10:38.418 Rx SBUS1_APPL.Slave.Lower SBUS_INTF.Data_Ind
18 12:10:38.418 Tx SBUS1_APPL.Slave.Lower SBUS_INTF.Data_Rq
19 12:10:38.418 Rx SBUS1_APPL.SBUS.Upper2 SBUS_INTF.Data_Rq
20 12:10:38.418 Tx SBUS1_APPL.SBUS.Upper1 SBUS_INTF.Data_Ind
13 12:10:38.418 Tx SBUS1_APPL.Master.Lower
SBUS_INTF.Data_Rq
13 12:10:38.418 Tx SBUS1_APPL.Master.Lower
[HEX] : [0x00000000] 00 00 00 04 00 00 00 00
Rx: 81925 Tx:

SBUS1_APPL messages captured by the System Tracer

SHUT DOWN THE SYSTEM

To shut down the system completely, the SAFIRE Virtual Machine must be ended. To do this, simply close the SAFIRE Virtual Machine window. The System Tracer window will close automatically.

Copyright © 1990-2007 SOLINET GmbH. All rights reserved.
All product names are trademarks of their respective owners.
Specifications subject to change without notice.